Static GPU threads and an improved scan algorithm

Jens Breitbart

Research Group Programming Languages / Methodologies
University of Kassel

Ischia - Naples, Italy
August 30th, 2010
UCHPC 2010

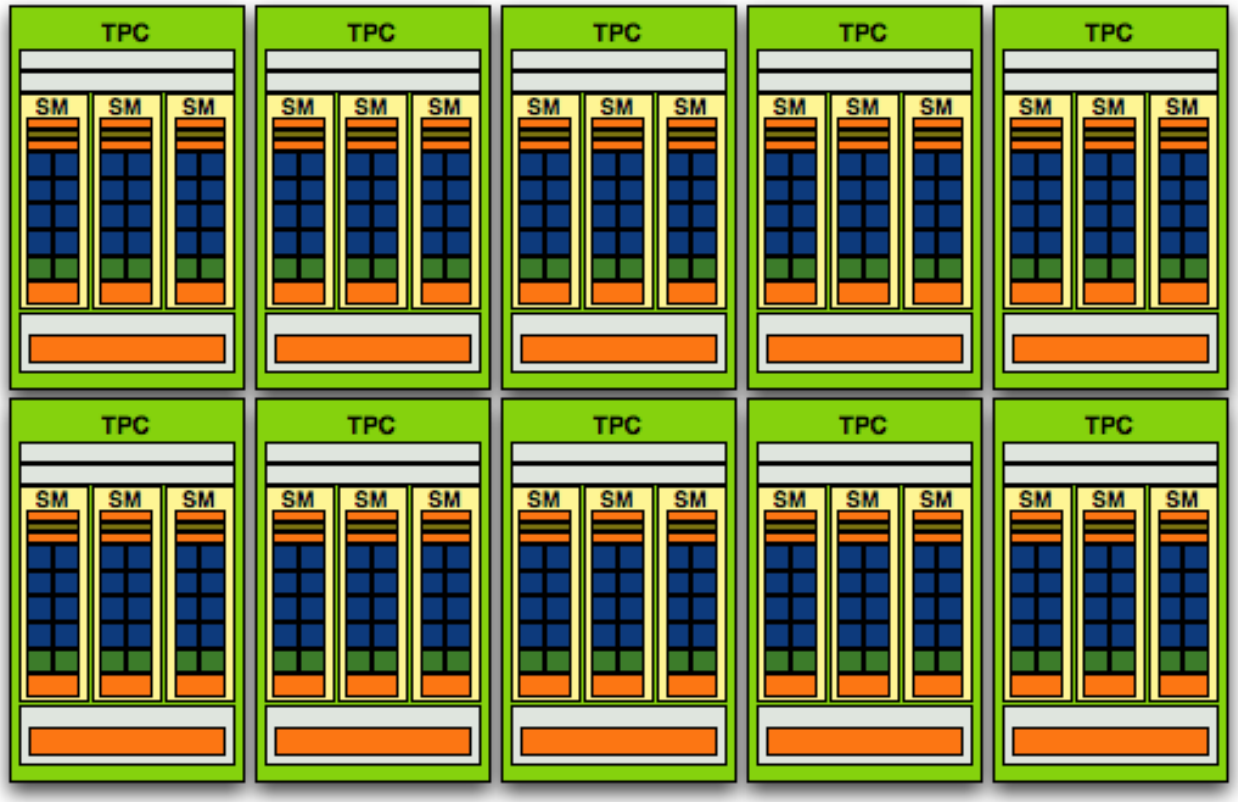
Goal

Is using

- global barrier synchronization.
- manual work distribution + task interleaving.

in CUDA/OpenCL worth the try?

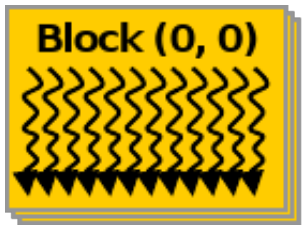
GPU Architecture



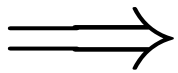
- GPUs are tile based many core architectures.
- Communication between tiles must be done using off-chip memory

CUDA/OpenCL mapping to GPU

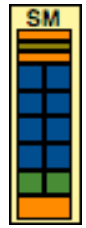
CUDA / OpenCL



Unlimited amount of workgroups



Hardware

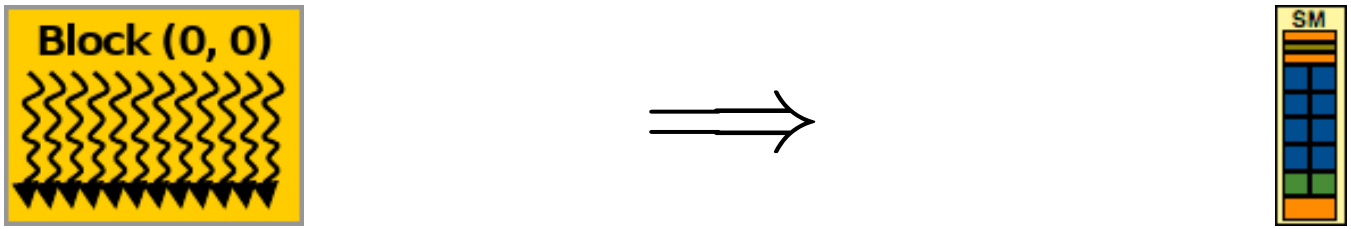


30 tiles (GT 280)

Synchronization between groups

Not all groups can be active at the same time, so **synchronization between groups is disallowed in CUDA / OpenCL**.

Static GPU threads

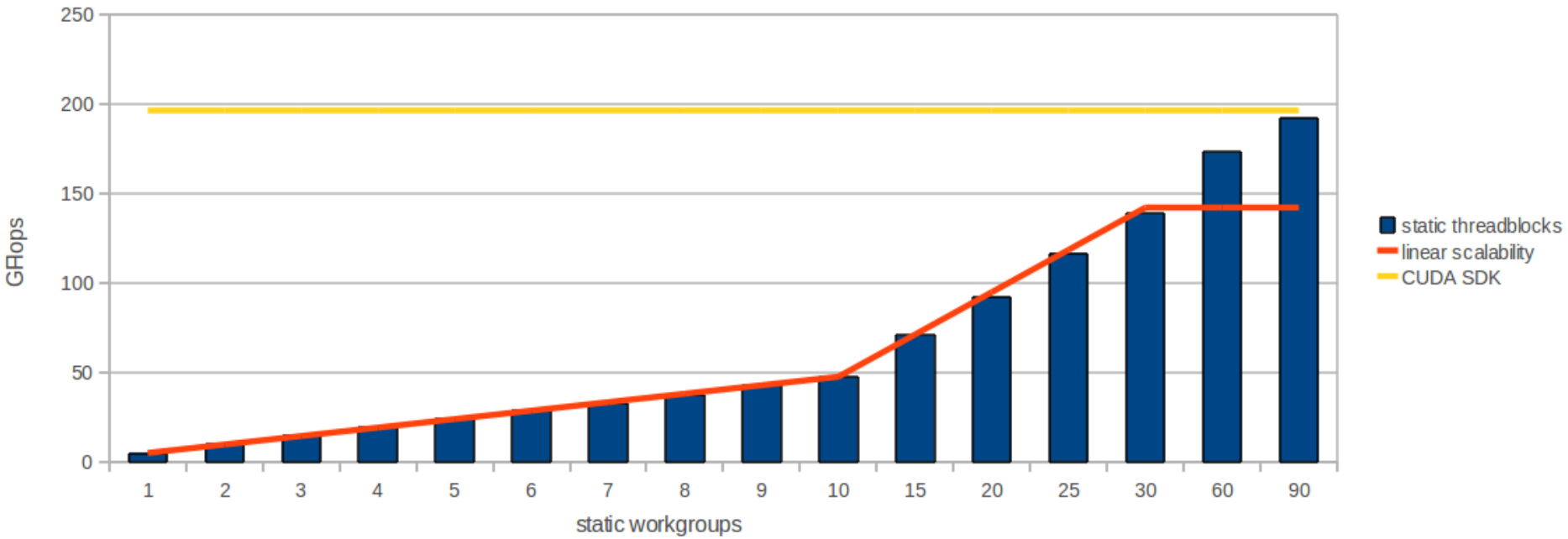


User only $\sim$ 30 workgroups and manually distribute task to the groups.

SM thread $\sim$ CPU thread

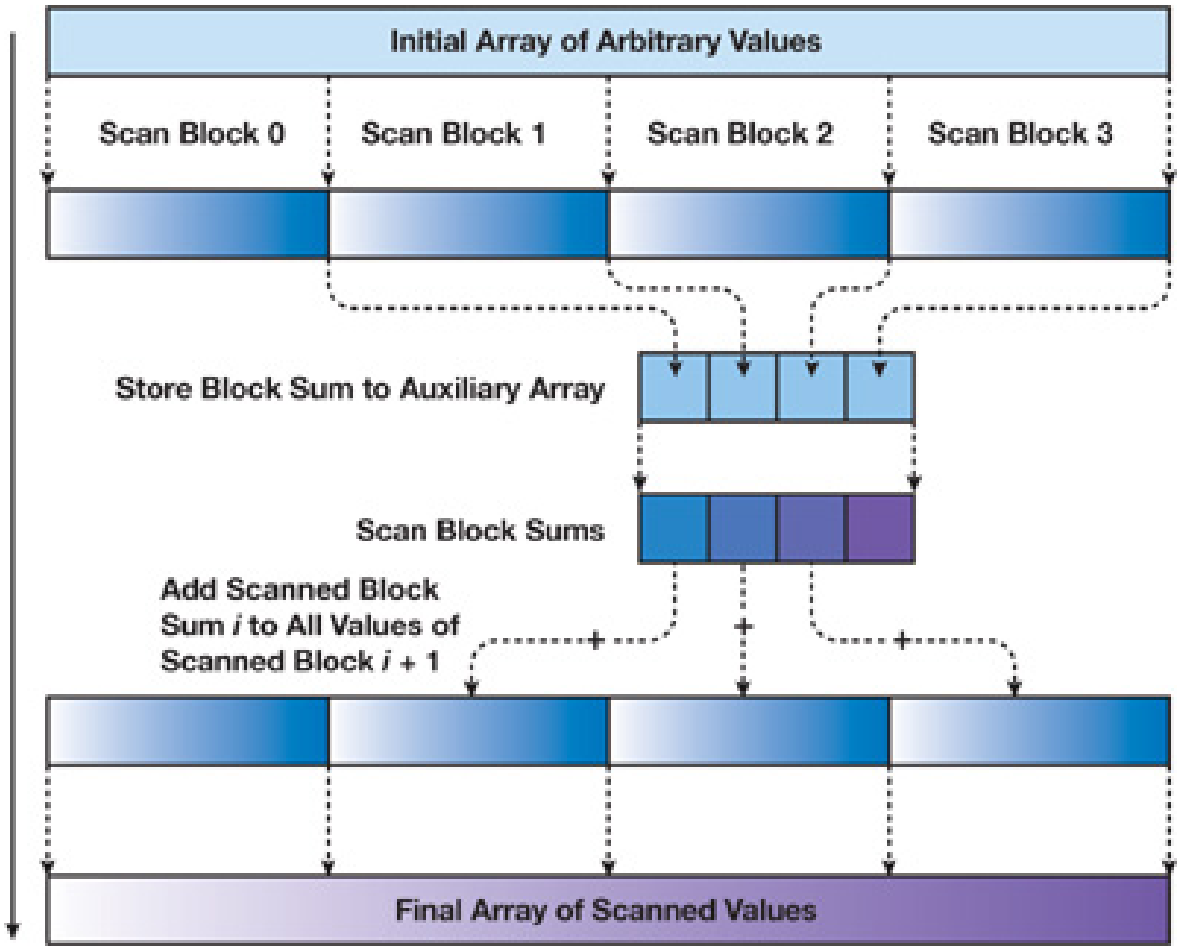
Think about a workgroup as an SM thread. An SM threads is used for multiple tasks – as CPU threads are.

Manual Work Distribution Overhead



- There is hardly any overhead compared to a “native” CUDA implementation.

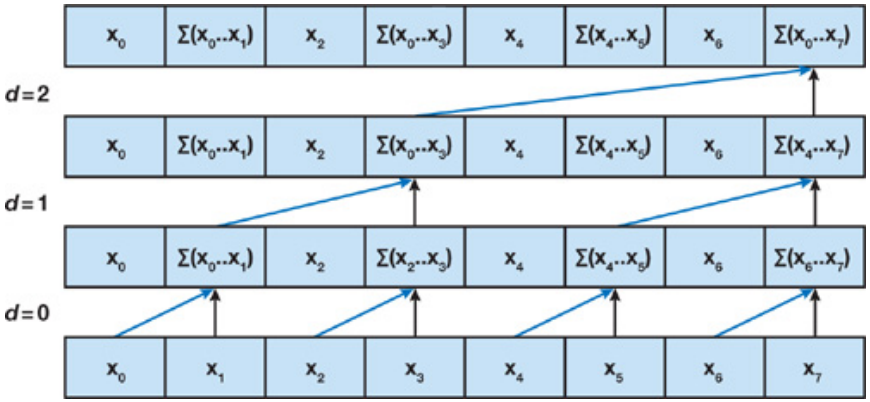
Scan algorithm by Harris et al. - Overview



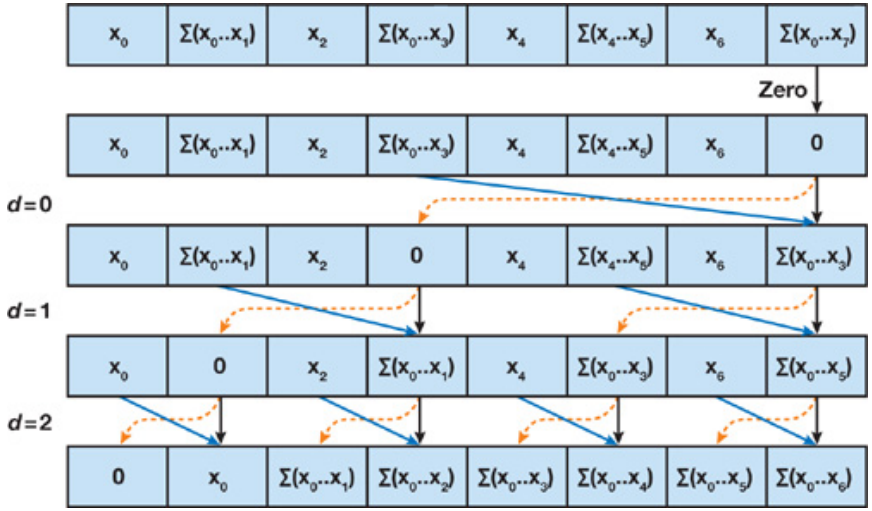
Source: `http.developer.nvidia.com/GPUGems3/gpugems3_ch39.html`

- The algorithm requires at least 3 kernel calls

Scan algorithm by Harris et al. - local scan



Up-sweep

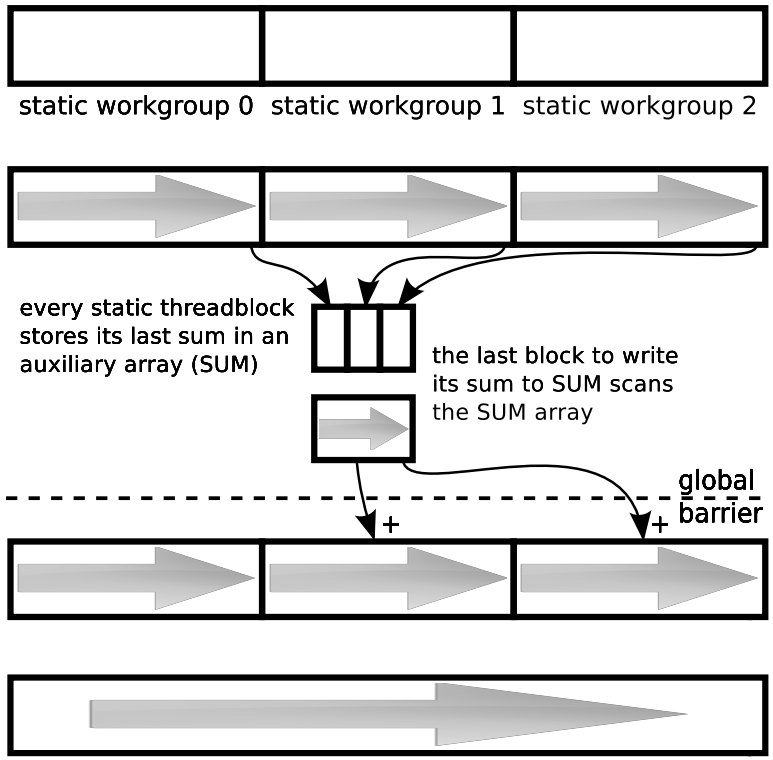


Down-sweep

Source: `http.developer.nvidia.com/GPUGems3/gpugems3_ch39.html`

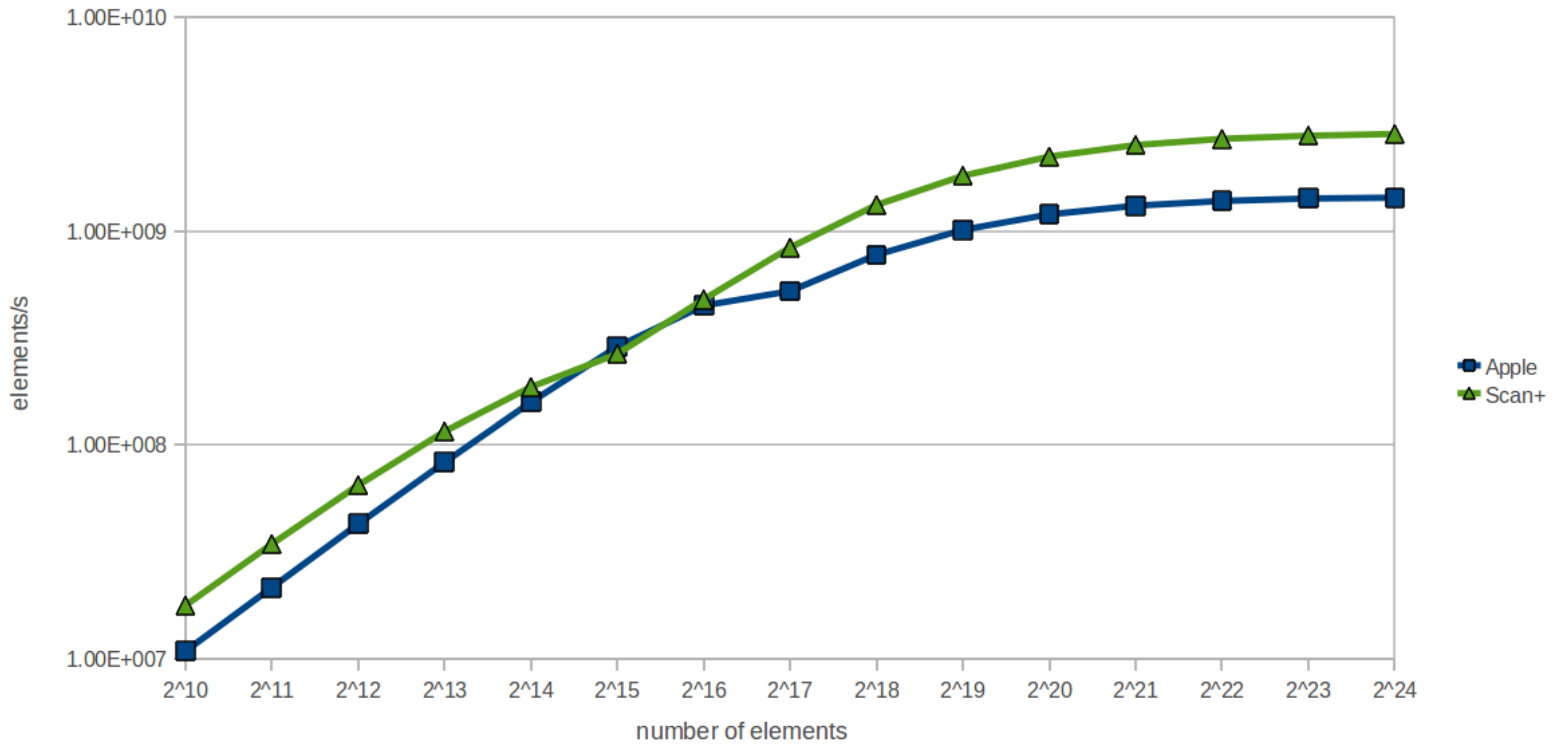
- Both phases are executed after one another.
- Only a fraction of the threads are active most of the time.

Improved scan



- Improved scan algorithm uses global barriers and task interleaving to increase thread occupancy.

Improved scan - performance



- About twice the performance of Appel's implementation of Harris algorithm.

Conclusion

- Depending on your algorithm you may get a decent performance improvement.
- There is hardly any runtime overhead using static threads.
- Writting your own barrier, work distribution code, ... is not trivial.

Conclusion

- Depending on your algorithm you may get a decent performance improvement.
- There is hardly any runtime overhead using static threads.
- Writting your own barrier, work distribution code, ... is not trivial.

Thank you

☺